Survivability in Wireless Networks: A Case for Overhead Reduction

Axel W. Krings
University of Idaho
Moscow, Idaho 83844-1010, USA

Abstract: *A link scheduling model is presented that utilizes primary-backup scheduling for packet scheduling. The advantage of this scheduling paradigm is that overhead can be suppressed in the fault-free case and overhead only needs to be endured in case of actual faults. The scheduling paradigm significantly increases survivability and can be used to reduce overhead of redundancy-based approaches. The foundation for using primary-backup scheduling in networks is derived. The schemes presented are very effective for multi-path protocols and MIMO and can be applied where watchdog-based algorithms fail or where geographic-centric disruptions render local approaches useless.*

1 Introduction

With the tremendous growth of wireless applications in recent years comes great concern for the lack of reliability, security and survivability. Especially in applications in the area of ad hoc and sensor networks there are many new challenges due to their features and the inherent characteristics of wireless technology. The main considerations have been routing and the overhead resulting from dealing with disruptions of the communication paths. As a result, many protocols have been introduced. However, in critical applications operating in hostile environments the security and survivability requirements may be much higher than usual and fault assumptions should include pathological behavior capable of introducing value faults. Furthermore, most research has focused on operation in benign environments where security considerations were not the driving motivation.

Since this work relates to tolerance of faults of different types under possibly pathological scenarios, we need to explore redundancy mechanisms. As such, any approach utilizing multipath and multiflow communication could have the potential for tolerating faults, if these concepts are exploited for reliability [12]. Many multipath and multiflow approaches have been presented in the literature, but their focus has not been on tolerating diverse faults but have rather been limited to overcome benign link or node faults. For example, the concept of multiflow has been used in [13] in the context of QoS enhancement, however, the focus is on transmission congestion. Multipath routing has been used to increase end-to-end reliability, e.g. the MP-DSR protocol in [8] forwards outgoing packets along multiple paths that are subject to a particular end-to-end reliability requirement; this however raises overhead concerns.

Primary and backup communication paths are considered in [9]. However, disjoint paths are not exploited for data redundancy but discarded as unwanted overhead. In their use of redundant disjoint paths the overhead to resilience tradeoff becomes unfavorable for a larger number of paths [2, 10]. We consider a different approach to primary and backup communication adopted from fault-tolerant multi-processor scheduling with focus on overhead reduction.

2 Network Survivability Model

For our purposes, the term survivability and reliability may be interchanged. Survivability was elected to emphasize that the operating environment may be malicious.

The communication network is represented as a digraph $G = (V, E)$, where computational nodes are the vertices and communication "links" are the edges. An edge e_{ij} is present in E if node v_j receives the signals of node v_i. If a source node v_S wants to establish a communication path with a destination node v_D, then the reliability of the path v_S-v_D is clearly depending on the reliability of the nodes and communication links along the path. To tolerate faults, may they be of benign nature or maliciously induced, one can chose to increase the survivability of the primary communication path v_S-v_D, e.g. using schemes such as presented in [11] or [6], or one can use a multi-path approach, considering alternative paths under the assumption that a certain threshold of "good" paths can mask faults.

Adding path diversity to the communication scheme, one inherits the undesirable overhead of multi-path routing and packet redundancy. In order to reduce overhead, we revert to using a proven mechanism from real-time scheduling. Specifically, we adopt primary-backup (PB) link scheduling, which was introduced in the context of fault-tolerant scheduling in real-time multiprocessor systems [1, 4]. Essentially, non-preemptive computational tasks (consisting of a primary and a backup task) are accepted into the real-time system if a feasibility test guarantees that the task can be scheduled to meet its deadline. Otherwise the task is rejected. If the primary task fails, due to transient or permanent faults, the backup task is executed. To avoid unnecessary overhead in the non-fault case, backup overloading is utilized. Whereas multiprocessor scheduling considers scheduling tasks onto processors, we are concerned with scheduling packets onto communication links. As such, a communication link and a processor are analogous. Similarly, data packets and computational tasks are analogous.

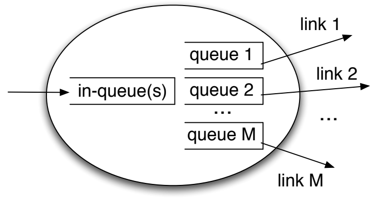

Figure 1: Conceptual Network Node

To make the analogy between links and processors some justification is necessary. We view a network node v_i of G as having separate links, i.e. channels, as shown in Figure 1. Packets are received into one or more input queues and scheduled on links via their associated output queues. This makes perfect sense in fixed networks, but in wireless nodes this view is only conceptual. Only in the case of MIMO (multiple-input-multiple-output), where dual-array multiple-antenna systems are used, is this representation apparent. However, in the absence of MIMO, we can still justify this view using multiplexing. For example, consider code division multiple access (CDMA). In CDMA multiple channels are multiplexed without dividing up the channel by time, thus logically implementing the concept of Figure 1. Time division multiple access (TDMA), on the other hand, allows multiple links to be emulated by sharing the link in a time-division scheme. Again, assuming the time slots are relatively small, the concept in the figure is preserved.

Next, we introduce notation for scheduling packets on links, or practically, their associated queues. Given the abstraction of a wireless node above, let L_j denote link j. We will speak of "scheduling packets on links", which actually means that packets are scheduled in the respective queues. Associated with each data *packet* P_i are the attributes *arrival time* a_i, i.e. the time at which P_i enters the in-queue of the node, the *ready time* r_i, which is the time the packet is ready to be moved to the outgoing link queue, the *start time* s_i, the time the packet is starting to be transmitted, *transmission time* l_i, which is the time it takes to send out the packet of size l, the *finish time* f_i, the time the last bit of the packet has left the link, and the *deadline* d_i, which defines the latest deliver time as needed to guarantee QoS. Note that $l_i = f_i - s_i$.

For each packet P_i a *primary* Pr_i and a *backup* copy Bk_i are defined. Note that "copy" can refer to redundant pointers to a single data object. The purpose of Bk_i is that, if the transmission of Pr_i fails, it will serve as a backup. The deadline for the acknowledgment of the primary's delivery in the fault-free case is called *acknowledge time*, $ack(Pr_i)$. Thus $ack(Pr_i)$ constitutes the maximal time up to which one can wait for an acknowledgment. The actual time when Pr_i is acknowledged is denoted by $t_{ack}(Pr_i)$, with $t_{ack}(Pr_i) \leq ack(Pr_i)$ in the fault-free case. Thus, if an acknowledge of delivery has not been received by $ack(Pr_i)$, it is assumed that a fault has occurred. However, if Pr_i is successfully delivered, which would be confirmed at $t_{ack}(Pr_i) \leq ack(Pr_i)$, then Bk_i can be discarded from the queue. The backup only requires link resources if the primary fails. Otherwise, the only penalty for utilizing the backup is the overhead associated with queue management. From a practical point of view, the value for $ack(Pr_i)$ is chosen based on the expected transmission time in the no-fault scenario. If the expected time it takes to acknowledge Pr_i is t_a, then $ack(Pr_i) = s(Pr_i) + \alpha t_a$ where $\alpha \geq 1$ is a constant affecting how sensitive the fault detection is. This should be only an expected (pessimistic) value, and thus high accuracy in a minimal $ack(Pr_i)$ may not be meaningful.

An acknowledge $t_{ack}(Pr_i)$ of a packet P_i addresses the round-trip delay of the packet, i.e. the time to deliver the packet plus the time it takes to send and deliver the acknowledge back to the sender. We will assume that the only way we can practically expect that a packet is delivered is at the time of its acknowledge $t_{ack}(Pr_i) \leq ack(Pr_i)$. This way we avoid the issues associated with the case where faults occur during the time of transmission or acknowledge. Note that $ack(Pr_i)$ is a parameter reflecting the expected transmission time in the absence of faults. This should not be confused with timeout parameters of the transport control protocol, e.g. TCP

The packet attributes defined for P_i above will be used for Pr_i and Bk_i as well, e.g. $s(Pr_i)$ is the primary's starting time or $f(Bk_i)$ the finishing time of the backup. We assume that in the schedule of packet P_i the timing relationship between Pr_i and Bk_i is $a_i \leq r_i \leq s(Pr_i) < f(Pr_i) \leq ack(Pr_i) \leq s(Bk_i) < f(Bk_i) < t_{ack}(Bk_i) \leq d_i$. Furthermore, we assume that if Pr_i fails, then backup Bk_i will succeed. Thus, at most one fault is assumed for packet P_i. An important assumption for PB scheduling is that the primary and backup of P_i cannot be scheduled on the same link, i.e. $L(Pr_i) \neq L(Bk_i)$.

In order to minimize the overhead associated with scheduling backup packets the concept of backup overloading is adopted. Figure 2 shows the concept. Packet P_1 has its primary Pr_1 scheduled on link L_1 and its backup Bk_1 on L_2. Similarly, P_2 has Pr_2 scheduled on L_3 with its backup Bk_2 on L_2, thus overloading L_2 from $s(Bk_2)$ to $f(Bk_1)$. This has consequences for the assumptions about faults.

In the figure both backup packets overlap. It can easily be shown that if two backups Bk_i and Bk_j are overlapping on a link, then their respective primaries must be scheduled on different links. Conversely, if Pr_i and Pr_j are scheduled on the same link, then their backups must not overload.

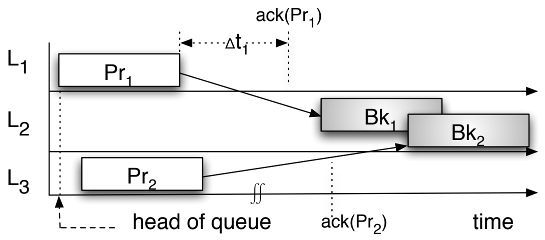

Figure 2: Backup Overloading

The desirable feature of backup scheduling is that given packet P_i, backup Bk_i can be deleted if Pr_i is delivered successfully at $t_{ack}(Pr_i) \leq ack(Pr_i)$. The usage of backup overloading requires the introduction of the notion of *Time to Second Fault* (TTSF), which is the time at which a second fault can occur without risking the loss of a packet due to overloading. Note that the smaller TTSF is, the more resilient the system becomes to second faults. Let TTSF(L_i) indicate the time to second fault with respect to link L_i. It can be shown that TTSF is the maximum TTSF(L_i), which is defined as the maximum time of the acknowledge of the primary whose backup is scheduled on the link and the acknowledge of the backup scheduled on the link.

Whereas the previous discussion considered benign and omission faults, we now turn to the impact of value faults, i.e. the case where the content of a packet is manipulated. To tolerate k such faults, by definition, one needs $2k + 1$ redundant packets, which will guarantee that the good packets are in the majority. This should not be confused with the Byzantine majority of asymmetric faults in distributed agreement [7].

If one wants to detect a single value fault using PB scheduling one can extend the concept to include two primary copies and a backup. Thus, for packet P_i we consider primary Pr_i, secondary Se_i and backup Bk_i. The deadline for the acknowledge of both Pr_i and Se_i is assumed to be $ack(Pr_i)$. Upon acknowledgment of both

Pr_i and Se_i the backup Bk_i is unscheduled. Conversely, if either Pr_i or Se_i fail to acknowledge, Bk_i is required. It should be noted that in principle scheduling of a primary and a secondary on disjoint links allows for correction in the case of a benign and omission fault and for detection of a value fault. In the latter case, the possible tie between packets can be resolved with the backup packet, constituting fault recovery. Thus, logically this scheme corresponds to the so-called hybrid-SCP-TMR [3], where in the case of real-time multi-processor scheduling two copies execute first, implemented as a Self Checking Pair (SCP). If the outputs do not agree, the third copy is scheduled to break the tie, thus implementing Triple Modular Redundancy (TMR).

In the context of link scheduling, the detection mechanism of the hybrid-SCP-TMR requires further explanation. Note, that by the definition of this configuration the detection of a value fault requires that a difference in the packet contents must be observed. In the multiprocessor case of [3] this is done by a comparator, e.g. a voting task, which detects that the results of the two tasks differ. In the network protocol stack the detection of differences in the packet contents can be observed by the receiver of the packets, e.g. by the observation that the signatures (or frame check sequences) of the primary and secondary packets do not match.

If the receiving node detects that the content of Pr_i and Se_i do not match, then an explicit or implicit message $reject_i$ is issued. An explicit reject message identifies the mismatch of the packet content between the two copies of P_i. Alternatively, an implicit reject is realized by simply not acknowledging a packet, thus triggering a timeout at $ack(Pr_i)$. In both cases backup Bk_i is sent to break the tie.

Next, we want to establish the timing relationships of the packets. Assuming $s(Pr_i) \leq s(Se_i)$, the timing relationship between Pr_i, Se_i and Bk_i is $r(P_i) \leq s(Pr_i) \leq s(Se_i) < ack(Pr_i) \leq s(Bk_i) < f(Bk_i) < t_{ack}(Bk_i) \leq d_i$. Furthermore, we have $f(Pr_i) \leq ack(Pr_i)$ and $f(Se_i) \leq ack(Pr_i)$.

To avoid packet loss in the presence of a permanent value faults the primary, secondary and backup of a packet must be scheduled on different links, i.e. $L(Pr_i) \neq L(Se_i) \neq L(Bk_i)$. This follows directly from the function of a TMR, which can handle exactly one value fault under the assumption of independence of faults. Scheduling two or more copies of the packet on the same link would violate this independence assumption.

As in simple PB scheduling we assume that if Pr_i or Se_i fail, i.e. one packet content is corrupted, then backup Bk_i will succeed. Assume that packets P_i are scheduled using backup overloading under a hybrid-SCP-TMR strategy. Furthermore, assume that at time t link L_k experiences permanent value faults. Then another fault can be tolerated at time $t' = \max\{t_1, t_2, t_3\}$, where

$$t_1 = \max\{t_{ack}(Bk_i),\ \forall Pr_i : L(Pr_i) = L_k\}$$

$$t_2 = \max\{t_{ack}(Bk_i),\ \forall Se_i : L(Se_i) = L_k\}$$

$$t_3 = \max\{t_{ack}(Pr_i), t_{ack}(Se_i),\ \forall Pr_i, Se_i : L(Bk_i) = L_k\}$$

If the exact time of $t_{ack}(Pr_i) \leq ack(Pr_i)$ is not known, $t_{ack}(Pr_i) = ack(Pr_i)$ must be assumed. The same holds for Se_i and Bk_i. Contrary to the case of a simple primary-backup scheme, now the overhead associated with the secondary has to be tolerated. However, overhead induced by the backup packets are still suppressed in the non-fault case.

3 Reliability Analysis

The reliability of a communication channel, $R(t)$, is the probability that the communication is failure-free during the entire time-interval $[0, t]$. In order to determine the reliability of communication using PB link scheduling four approaches were analyzed, assuming fail rate $\lambda = 10^{-3}$ per time unit. First, a Single Path was considered, i.e. a communication path without packet redundancy. Second, simple PB scheduling was considered, however, we relaxed the assumptions about a guaranteed delivery of the backup packet, since in a real system no such guarantee can be given. Thus, the results shown are more realistic, but at the same time more pessimistic. Third, we considered Hybrid SCP-TMR scheduling for value faults, again under the relaxation of guaranteed backup packet delivery. Fourth, we used the previous scheme, but only considered benign faults. This effectively changes the hybrid SCP-TMR into a 1-of-3 system. The results of the four different approaches are shown in Figure 3. As can be seen all primary-backup approaches show significant

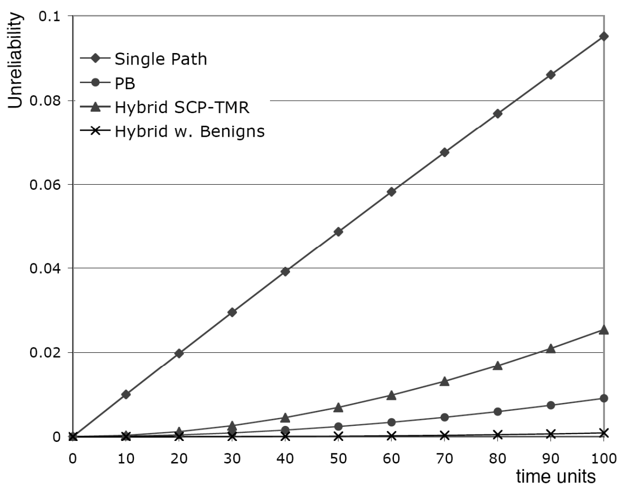

Figure 3: Unreliability for PB Scheduling

improvements over the single path approach. Furthermore, in non-faulty scenarios the improvements come at no communication cost.

4 Conclusions

Primary-backup link scheduling was introduced as a mechanism that significantly increases survivability for wireless networks. The concept was demonstrated for multi-path networks for simple fault models (benign and omission faults) and value faults. The overhead associated with the scheme results in only negligibly small local link scheduling overhead in the fault free case. Thus, the burden of multi-path packet overhead was only induced if an actual fault occurred.

References

[1] R. Al-Omari, et.al, Efficient overloading techniques for primary-backup scheduling in real-time systems, *Journal of Parallel and Distributed Computing*, Vol. 64, Issue 5, pp. 629-648, May 2004.

[2] D. Ganesan, et.al., *Highly-resilient, energy-efficient multi-path routing in wireless sensor networks*, Mobile Computing and Communications Review, Vol. 4, No. 5, October 2001.

[3] O. Gonzalez, et.al., Adaptive Fault Tolerance and Graceful Degradation Under Dynamic Hard Real-time Scheduling, *Proc. IEEE Real-Time Systems Symposium*, pp. 79-89, 1997.

[4] S. Ghosh, et.al., Fault-Tolerant Scheduling on a Hard Real-Time Multiprocessor System, *Proceedings of the International Parallel Processing Symposium*, pp. 775-782, 1994.

[5] S. Ghosh, et.al., Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems, *IEEE Trans. Parallel Distributed Systems*, 8 (3), pp. 272-284, March 1997.

[6] A. Krings, and Z. Ma, *Fault-Models in Wireless Communication: Towards Survivable Ad Hoc Networks*, Military Communications Conference, MILCOM 2006, pp. 1-7, 23-25 Oct. 2006.

[7] L. Lamport, et.al., *The Byzantine Generals Problem*, ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, pp. 382-401, July 1982.

[8] R. Leung, J. Liu, E. Poon, A. Chan and B. Li, MP-DSR: A QoS-aware Multi-path Dynamic Source Routing Protocol for Wireless Ad-Hoc Networks, *Proc. 26th Annual IEEE Conference on Local Computer Networks*, LCN 2001, pp. 132-141, 2001.

[9] H. Liu and D. Raychaudhuri, *Label Switched Multi-path Forwarding in Wireless Ad-Hoc Networks*, Proceedings of the 3rd Intl Conf. on Pervasive Computing and Communications Workshops, (PerCom 2005 Workshops), pp. 248-252, 2005.

[10] M. K. Marina and S. R Das, *On-Demand Multipath Distance Vector Routing for Ad Hoc Networks*, Proc. of the International Conference for Network Protocols (ICNP), Riverside, USA, pp. 14-23, 2001.

[11] Sergio Marti, et.al., *Mitigating routing misbehavior in mobile ad hoc networks*, Mobile Computing and Networking, pp. 255-265, 2000.

[12] S. Mueller, R. P. Tsang, and D. Ghosal, *Multipath Routing in Mobile Ad Hoc Networks: Issues and Challenges*, MASCOTS 2003, LNCS 2965, Springer-Verlag, pp. 209234, 2004.

[13] N. Thanthry, et.al., *TCP-M: Multiflow Transmission Control Protocol for Ad Hoc Networks*, EURASIP Journal on Wireless Communications and Networking, Article ID 95149, 16 pages, 2006.